Transformer Models for Enhancing MirrorGAN Based Text to Image Generation

Dinghao Shen
EECS Department
University of Michigan
Ann Arbor, USA
jackyy@umich.edu

Yuan Yao
EECS Department
University of Michigan
Ann Arbor, USA
kelyao@umich.edu

Ying Yuan
EECS Department
University of Michigan
Ann Arbor, USA
yyyuan@umich.edu

I. Introduction

What is the task? A description of the particular problem(s) you are addressing. The motivation behind the project and why this is an important problem to study.

Text-to-image generation (TTI) is a useful and challenging task of generating an image from the given text description and has great potential in various fields, such as scene retrieval, computer-aided design, and entertainment in people's daily life. The successful demonstration of TTI generation would also be a milestone in artificial intelligence. However, based on the generation quality of the current deep learning models, existing techniques could not yet be utilized well in real-life applications.

Up to now, a number of Generative Adversarial Network (GAN) based deep learning methods have been developed for TTI generation [1]. MirrorGAN [2] is a popular method that exploits the idea of learning TTI generation by redescription. Meanwhile, pretrained transformer model is proved to be successful in text feature extraction [3], which is an essential part of TTI generation models. This motivates our project idea. We decide to focus on amplifying the effect of text feature extraction using transformer in the TTI generation model as a direction to further improve the generation quality. We aim to investigate the use of state-of-the-art pretrained transformer models, such as BERT [3], within MirrorGAN, a representitive TTI generation model. [1]

What are the challenges of the task?

Challenge exists in correctly implementing the model. Language is always sequential, while the TTI generation is multimodal [4]. This difference and the complexity of MirrorGAN make difficulty in combining BERT and MirrorGAN.

Challenge exists in reducing the training cost, as well as efficient optimization of the model. The complication of TTI generation model and the large datasets resulting in the high training cost and long training time. MirrorGAN alone involve the training of 3 seperate modules. The complexity in combining BERT and MirrorGAN increases the training workload for hyperparameter tuning and joint optimization.

[1]Code available at https://github.com/item084/eecs592-proj

Challenge also exists in evaluation. Existing quantitative evaluation metrics are far from perfect for TTI generation. As far as we know, there is a huge gap between quantitative metrics and qualitative evaluation using human judgment in works like AttnGAN and MirrorGAN. We want to explore the potential of newly proposed metrics like FID in mitigating this issue.

System input and output. What is the input that the model receives? What is the output that the model generates?

Examples could be seen in Fig. 1. The text description is the input and the generated image is the output.

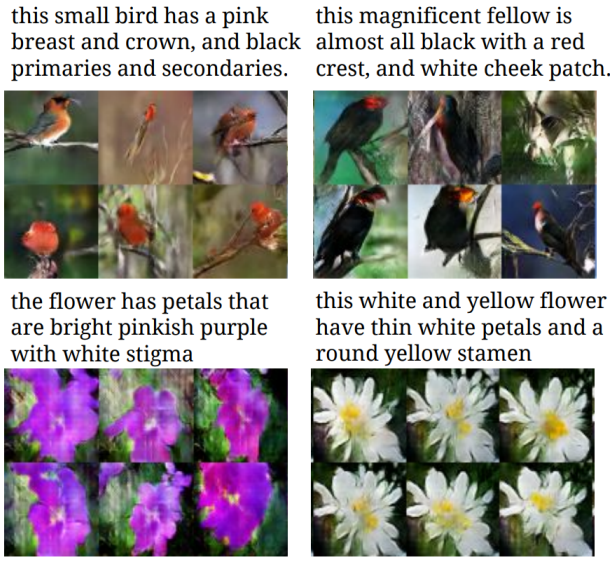

Fig. 1. Examples of the images generated from text description.

What are your contributions in the project? Give a brief overview of what you have done and achieved.

We combined the techniques in a new architecture, trained the models, and evaluated their results with new evaluation metrics, Fréchet Inception Distance (FID) [5]. We reproduced AttnGAN, MirrorGAN and compared their performance with our newly proposed model. We faced challenge in correctly implementing the model due to the high complexity of MirrorGAN. The training imposed high demand on GPU and

memory which we managed to reduce with the use of pretrained BERT model.

The result proves the feasibility of combining the advance in the transformer-based model, BERT, and the TTI generation model, MirrorGAN. The improvements is also reflected in qualitative and quantitative evaluations. Our model generates images with better textual details and backgrounds with lower training cost.

Report the task allocations among your team members.

Yuan Yao and Ying Yuan reproduced the original MirrorGAN model and implemented the new model; Yuan Yao, Dinghao Shen, and Ying Yuan evaluated the results of original MirrorGAN model and the new model. As far as we know, our team members contribute equally and collaborate well throughout the project.

II. Related Works

Put your work in context: What has been done before related to your topic? Please make sure that you cite relevant and (if possible) relatively new research.

These years, Recurrent Neural Network (RNN) architecture is ideal for extracting the semantic information from text description. Other than RNN-based model, the transformer-based model makes further improvement for NLP [3]. At the same time, GAN architecture has begun to generate fake images that are indistinguishable from real photographs. In 2016, TTI generation reached a new height by bridging the RNN and GAN to translate from characters to pixels [1]. This GAN-based model is able to generate photo-realistic images from simple text descriptions.

Since then, a number of GAN-based deep learning methods have been developed for TTI generation [1] and have made great progress. AttnGAN enables the algorithm to creating different sections of the image based on different sections of the text. AttnGAN also measures the similarity between the text and the generated image using a deep attentional multimodal similarity model (DAMSM) [6].

MirrorGAN utilizes the idea of learning text-to-image generation by redescription and conducts supervised learning by using paired text-image data [2]. It consists of three main modules to generate the image: a semantic text embedding module (STEM), a global-local collaborative attentive module for cascaded image generation (GLAM), and a semantic text regeneration and alignment module (STREAM) [2]. First, it turns the text description of the image into a semantic text embedding using STEM. Then, in GLAM, the word embedding and visual feature are fed into cascaded image generation leveraging both local word attention and world sentence attention to generate images. Finally, it aligns the text redescription from the generated image with original text description in STREAM [2] to compare with the original text. MirrorGAN improves the performance of text-to-image generation by redescription.

Another revolutionary model to be considered is Bidirectional Encoder Representations from Transformers (BERT). BERT is a language representation model using two unsupervised tasks, Masked Language Model (MLM) and Next Sentence Prediction (NSP), to capture word-level and sentence-level representations, respectively [3]. It has been proved to be successful in many applications of NLP. BERT model can be finetuned with just one additional output layer to create models for a wide range of tasks [3]. Great achievements have been made with BERT.

Considering the magic of BERT, some scholars start to apply BERT to AttnGAN model. It's proved that combining BERT and AttnGAN does enhance the performance [7].

What's new in your work? Do you provide new analysis or insights which can be used for future model development?

We find out that MirrorGAN only utilizes a basic method for text embedding in STEM. To amplify the effect of semantic information extraction, we propose a MirrorGAN model variant that takes the advance of the state-of-the-art pretrained transformer models, BERT [3], with the redescription idea from MirrorGAN.

Moreover, STEM is pretrained on target dataset, which is unable to generalize and increases complication in training. Using BERT would significantly reduce the training cost.

For the evaluation process, MirrorGAN uses inception score to evaluate the model. Inception score takes only the generated images as input and measures the variety and authenticity of the generated images. Without the comparison with ground truth, it is difficult for the machine to simulate the human eyes to evaluate the generated images. We adopt the FID as a new metrics to measure the distance between real images and the generated images, in an attempt to mitigate the gap between numeral metrics and human judgment.

Insights for future work:

- Using pertrained language models as text encoder in TTI generation tasks reduces training cost, provides better generalization, without sacrifice much performance.
- Lack of encoded background information hurts the overall quality of generated image. We may need separate input channels for the object and its background for better performance.
- There is inconsistency between different metric (e.g. IS, FID, etc.) for TTI generation tasks. Properly interpreting and combining them helps mitigating the gap between the quantitative metrics and human's subjective judgment.

III. Methodology

A. Dataset

What data do you use? What is the size of it? What information is contained?

We are working with the Caltech-UCSD Birds-200-2011 (CUB-200-2011) dataset [8]. The CUB-200-2011 dataset contains **11,788** images categorized by 200 bird species. The images are obtained from Flicker image search by the researchers. Each image comes with 1 subcategory label, 15 part locations, 312 binary attributes and 1 bounding box. The image dataset is expanded by fine-grained natural language

descriptions. **10** single-sentence descriptions are collected for each image through the Amazon Mechanical Turk (AMT) platform. Each sentence is required to have least 10 words, without subcategories and actions information [9]. A sample data is shown in Fig. 2.

Fig. 2. A sample image and text annotation from CUB-200-2011 (ID: 542, Name: Red_Winged_Blackbird_0055_4345).

Are there any characteristics of the data?

Every image is focused on the bird as the only object in the image. The bounding box is placed so that the bird is in the center. The background are diversified and are not focused in many of the images. The text annotations mainly describe the colors in different body parts of the bird. Other descriptions include the shape of the beak, the shape of feet, and sometimes the overall size of the bird. There is no description on the background.

Why is it suitable for your task?

The dataset is widely used for multi-class categorization and part localization tasks. It is also used for text-to-image generation tasks in StackGAN, AttnGAN, and MirrorGAN.

The dataset is suitable for our projects partly because it is relatively small in size, which make the training cycle a little shorter to fit in the time bound of a course project. It is used in previous works on text-to-image generation like StackGAN, AttnGAN, and MirrorGAN, so that we could reuse their implementation for partitioning and preprocessing, and evaluation. Thanks to this, we can make direct comparisons with the outcome of previous works.

B. Preprocessing

How do you collect and clean the data?

The dataset is downloaded from the link provided in the MirrorGAN's official git repo [17]. It expands the original dataset with a volcabulary on the annotations and text-id mapping of the words. It also has a partition of training and testing set, namely 8,855 for training and 2,933 for testing.

By strictly following the metadata provided by MirrorGAN, we do not further clean the data.

How do you process the data to fit into the structure that the model expects?

The preprocessing of the image data includes: 1) Applying the bounding box; 2) Resize the image and randomly crop and horizontally flip the image; 3) Normalize the image to get the tensor representation. Fig. 3 shows snapshots of different stages in the preprocessing for a sample image.

Fig. 3. Preprocessing of a sample image.

For any given image, the data after preprocessing has the size of $N \times N \times 3$, where RGB channels are retained. N could be 64, 128, or 256, based on which branch it serves in the GLAM architechture.

The processing of the text are based on the vocabulary built on the annotations provided by MirrorGAN. The size of the volcabulary is 5453 including 3 special tokens <start>, <end>, and <pad>. The conversion between texts and their vector representation is done by using a pair of python dictionaries.

C. Environments

In our current setting, our model is the only agent. Therefore, the environment, accordingly, is the input training images and captions as well as the input texts when testing. The complexity of the environment varies based on different dimensions. One on hand, it's deterministic because it only takes images and texts in the same format, which is under our control. However, on the other hand, it's complex because these images and captions have corresponding relationship between each other. And our agent should explore the connection. The interaction mechanism between the environment and our agent is simply our model takes images and texts data to "learn" itself, and generates images as output based on fed in captions. The environment is fully oberservable, single-agent, deterministic and static. The input-output structures are simply how the model takes captions and generates images, which would be discussed in the next subsection. The goal of the environment is both to provide images and texts for the agent (our model) to learn from, and to offer generated captions expecting well-generated images.

D. Model Architecture

What methods do you experiment with? And why do you think they are reasonable and suitable for the task? Please include the details of your model architecture.

The primary model for this study is MirrorGAN. We use the MirrorGAN model as the baseline, and implement a

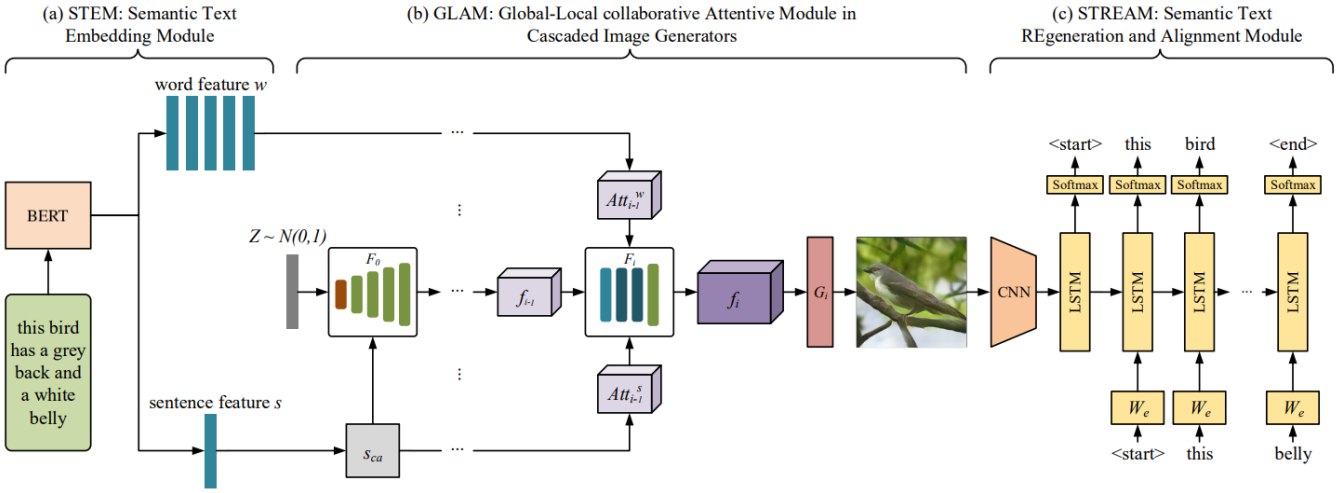

Fig. 4. Schematic of the variation of MirrorGAN with BERT Model [2]

novel model by modifying MirrorGAN with BERT. More specifically, we replaced the text-encoder of MirrorGAN with a pretrained BERT model.

The original architecture requires multiple pretrained models. None of them are publicly available. We tried to train our own pretrained models based on the evidence given in the released git repo. During the training we tried to improve their performances individually.

Our experiment methods involves: 1) optimizing the pretrained modules including the text encoder and the image-captioning module; 2) modifying the MirrorGAN model with a pretrained BERT; 3) compare the results qualitatively and quantitatively.

Details and justifications will be discussed later in this section.

1) Semantic Text Embedding Module (STEM):

STEM is used to provide the embedded text input in MirrorGAN. It borrows the idea of DAMSM from AttnGAN. The input to STEM are pairs of preprocessed images and descriptive sentences.

In AttnGAN, DAMSM consists of two networks, a Recurrent Neural Network (RNN) for text encoding and a Convolutional Neural Network (CNN) for image encoding. The RNN model extracts semantic embeddings from the text description. The CNN model maps images to semantic embeddings. An *attention-driven image-text matching score* is used to design the DAMSM loss in order to achieve the matching between text and image embeddings. Basically, it trains the encoders to provide text and images embeddings of better similarity, as an indication of better quality.

STEM in MirrorGAN only use the text encoder part of the pretrained DAMSM, which generates both local word-level embedding and global sentence-level embedding.

The purpose of using the pretrained encoder for the STEM module is to provide for the generative module a good enough text embedding. However, it involves additional complication of training that the *stability* is not guaranteed. Also, it is questionable whether similarity is a suitable substitution for the embedding *quality*. Further, The encoder has to be trained locally on the target dataset, which means it cannot be *generalized* to text-to-image tasks on different datasets.

Our idea is to replace the text encoder with a pre-trained BERT module. It is more stable because no extra training is required. It has high embedding quality that is well studied. It provides generalization because of the vocabulary it supports.

The original text encoder uses a **bi-directional LSTM**. The size of embedding vector is 300. The size of hidden layer is 128. There is 1 layer in each direction. The outputs of the two hidden states in each direction are concatenated to get the word features of W $\times$ 256. The output of the two last hidden states are concatenated to get the sentence feature of 256.

The new text encoder uses a pretained **uncase base BERT** downloaded from HuggingFace transformers library. No further finde tuning is involved. The output of the last hidden layer is used for the word features, which is of size W $\times$ 768. The BertPooler output of the [CLS] (class) token is used as the sentence feature, which is of size 768. The BertPooler is essentially a fully connected layer with a tanh activation.

After that, the word embeddings goes into the generator and the sentence embedding goes into the generator after conditioning augmentation [12].

2) Global-Local collaborative Attentive Module in Cascaded Image Generators (GLAM):

GLAM is a cascaded image generator that leverages both local word-level attention and world sentence-level attention at each stage. The inputs are from the STEM modules, and part of its loss is from the STREAM module. The idea is also brought from AttnGAN [7].

In each stage, it first feeds the word embedding (from the aforementioned text encoder) and the visual feature from the previous stage into a word-level attention model to generate an attentive word-context feature. Then, it uses a sentence-level attention model to enforce a global constrain in the same fashion.

The discriminators consists of a visual realism adversarial loss and a text-image paired semantic consistency adversarial loss, which are derived from fake image distribution, ground truth image distribution, and sentence level embedding.

In our project, we used the existing implementation from MirrorGAN with some minor bug fixes.

3) Semantic Text REgeneration and Alignment Module (STREAM):

STREAM is an encoder-decoder-based image captioning model that regenerates text descriptions from the generated images and aligns them the with original text descriptions to compare. The alignment is done by propagating its loss function back to the generator. In training, the input to STEM are pairs of preprocessed images and descriptive sentences. In testing, the input to STEM are pairs of generated fake images and the input sentences.

The architecture of STREAM is similar to that proposed in [13]. The encoder part of STREAM is a resnet-152 CNN model pretrained on the ILSVRC-2012-CLS image classification dataset. The decoder of STREAM is an LSTM. The hyperparameters include size of the embedding vectors, size of hidden layers, and number of LSTM layers. Our goal is to explore the property of this module and develop a better pretraining for the task. Details to be seen in later sections.

The overall diagram of the model is shown in Fig. 4.

IV. Experiments and Results

A. Dataset Partitioning

What is your dataset size? How do you split the dataset into training/validation/test sets?

We inherited the partitioning of the 11,788 images from AttnGAN and MirrorGAN's published data, which has 8,855 for training and 2,933 for testing. The partitioning information is included in the downloaded dataset.

We trained all modules using the training set of size 8,855.

B. Implementation Details

The released code base for MirrorGAN is incomplete and outdated in terms of the modules they used. We examined the code and modified them to enable training on the latest python3 and pytorch version. During our training, we identified and fixed multiple correctness bugs in the original implementation.

In addition, we modified the GAN module to support the use of pretrained BERT. We modified the image-captioning module to improve its performance.

We also implemented the evaluation methods with the help of existing codes.

1) Piecing together the model:

The published MirrorGAN repo did not provide the entire code base. Parts of the pretraining codes are scattered in other repos like AttnGAN's. We adapted our code based on the hints provided in the MirrorGAN's published repo file.

2) Fixing bugs:

We thoroughly examined the code and updated the outdated functions to the equivalent ones in newer versions of python and pytorch. During our examination of the code base, we identified several bugs that will affect the correctness of the implementation.

One of them is related to the randomized selection of annotations. For each image from the training set, the dataloader randomly selects a corresponding annotation out of 10 that are provided. The range of index should have been within 0...9 The random index generator in the code misuses RANDINT function and generate 0...10 instead. As a results, each image has a 1/11 chance to be associated with a wrong sentence. Somehow the bug did not cause a noticeable degradation in the entire model (assuming the same code was used to produce the results in the paper).

We also found mistakes in the dataloader in handing the special tokens. It seems to be mismatching <start>and <end>, and padding the short sentences with <start>. We suspect they made this mistake because they took the AttnGAN code without careful inspection. It could also be that they did not publish the final version of the code.

Other minor issues are also fixed as we further examined the code.

3) Implementing the usage of BERT:

The BERT encoder was implemented by replacing the pretrained text encoder with a pretrained BERT-base-uncased model from HuggingFace transformers library.

The additional effort in implementing the method involves the batching of sentences, and the handling of output of BERT. Values in configuration files are modified to fit the new module.

4) Modifying the image-captioning module:

By comparing with the original method proposed in [13], we found out that the implementation used in MirrorGAN lack some elements like the dropout layer. Also, the hyperparameters for the model are not specified. We fixed the issues in the code we use. Though, we did not get to study the effect of the missing dropout layer on the overall performance because of the time constraint.

5) Implementing the evaluation methods:

We adapted our evaluation methods from the existing implementations, and fit them to our data.

C. Training Process

We first trained the image-captioning module. Then, we trained the text-encoder. After that, we trained the generator and discriminator of the original implementation. In the end, we trained the generator and discriminator with the BERT modification applied.

1) Training the image-captioning module:

The problem with training the image-captioning module is that there is no clear boundary between underfitting region and overfitting region. The plateau is reached very quickly and lasts very long, as is shown in the plot Fig. 5.

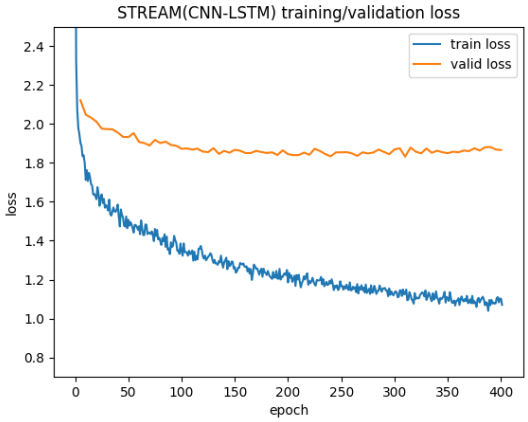

Fig. 5. CNN-LSTM training/validation loss

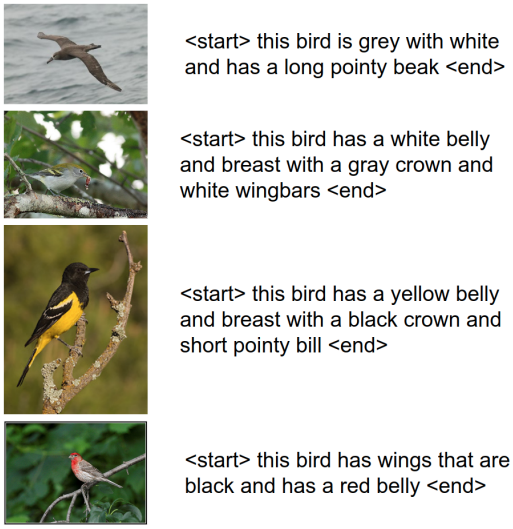

Fig. 6. CNN-LSTM captions example

We suspect that the model does not fit the data well. So we went on to test different hyperparameters and see if there is improvement. The tested hyperparameters include the embedding size, the hidden size, and the number of LSTM layers. We stuck with original batch size of 128 and learning rate of 0.001 because it gave the most steady loss curve with least noise.

We evaluated the result of training using BLEU scores [11], which measures the n-gram distance between the generated text and ground truth. The results are shown in the table below. We select the epochs with the lowest validation losses.

TABLE I
HYPERPARAMETER TUNING OF THE IMAGE-CAPTIONING MODULE.

embed	hidden	lstm	BLEU1	BLEU2	BLEU3	BLEU4
256	256	1	0.9328	0.7915	0.6009	0.3994
256	512	1	**0.9342**	**0.8110**	**0.6460**	**0.4669**
256	512	2	0.9312	0.7890	0.5975	0.3979
512	512	2	0.9328	0.8004	0.6206	0.4303
512	1024	2	0.9312	0.7924	0.6023	0.4034
512	1024	4	0.9330	0.7936	0.5981	0.3924

As we can see from the results, the model with embedding size of 256, hidden size of 512 and 1 layer of LSTM gives the best result. Although, the improvement seems to be marginal. A through grid search may generate more convincing results. But we did not get to do that because of the time constraint.

We also generated captions for real images as a sanity check.

2) Training the text-encoding module:

The training of STEM module using DAMSM also has a long-lasting plateau, which makes it hard to select the best epoch. It was trained using Adam optimizer with learning rate 0.0002.

We experimented with different batch sizes like 8, 16, 32, 48. We found that larger batch sizes gives more steady loss curves, meaning less noise in selecting an epoch of lowest validation loss. The trend of loss is shown in Fig. 7, where the red ones are losses associated with words and the blue ones are associated with the sentence. The selected epoch was 100.

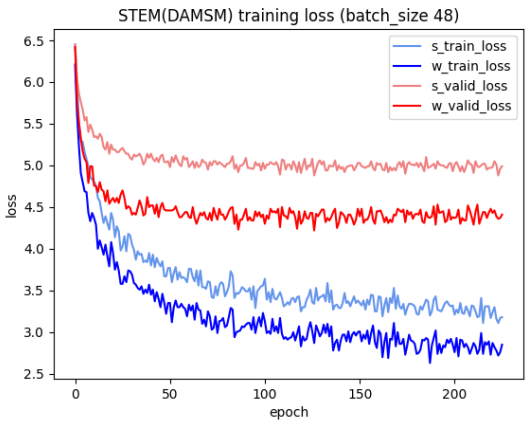

Fig. 7. DAMSM training/validation loss with batch-48

3) Training the generator and discriminator:

When training the GLAM, the weights of the two pretrained parts are fixed. The GLAM module was trained for 650 epochs using Adam optimizer with learning rate 0.0002. The same configurations are both used for the original and the BERT-modified version. Because it takes days to train the module,

we did little in the hyperparameter tuning. Evaluations are discussed in later parts.

D. Evaluation Metrics

Qualitatively, we evaluate the generated images with human eyes. Namely, we do subjective visual comparisons on the text input and the generated images to see if the images match the text. When evaluating the models with eyes, we focus on authenticity and semantic consistency of the generated images. Authenticity means whether the generated images look like real photographs with respect to birds and background. Semantic consistency means whether the generated images satisfy the descriptions of the captions with respect to color and shape.

Quantitatively, we use Inception Score and Fréchet Inception Distance to evaluate our model.

1) Inception Score: Inception Score (IS) was first introduced in *Improved Techniques for Training GANs* in 2016 [14]. Inception score takes only the generated images as input, measuring the variety of generated images and the authenticity of the generated images. Inception Score makes use of Inception V3 model, an image classification network from Google which takes images and returns probability distribution. We will later call it the Inception model for simplicity. To calculate IS, we apply the Inception model [15] to every generated image to get the conditional label distribution $p(y \mid \boldsymbol{x})$. Images that contain meaningful objects should have a conditional label distribution $p(y \mid \boldsymbol{x})$ with low entropy. This measures the authenticity (or objectiveness of the images). Moreover, we want to measure the variety of the images. If the generated images are diverse, then the model should give a high entropy for the marginal distribution $\int p(y \mid \boldsymbol{x} = G(z))dz$. Therefore, to evaluate both dimensions, IS is defined as $\exp\left(\mathbb{E}_{\boldsymbol{x}}\mathrm{KL}(p(y \mid \boldsymbol{x})\|p(y))\right)$ [14]. KL here is a statistics formula called the Kullback-Leibler (KL) divergence. The KL divergence is a measure of how similar/different two probability distributions are. For discrete probability distributions P and Q defined on the same probability space, $\mathcal{X}$, the relative entropy from Q to P is defined [16] to be

$$D_{\mathrm{KL}}(P\|Q) = \sum_{x\in\mathcal{X}} P(x)\log\left(\frac{P(x)}{Q(x)}\right)$$

which is equivalent to

$$D_{\mathrm{KL}}(P\|Q) = -\sum_{x\in\mathcal{X}} P(x)\log\left(\frac{Q(x)}{P(x)}\right)$$

Therefore, KL divergence is high when our distributions are dissimilar, namely each generated image has a distinct label and the overall set of generated images has a diverse range of labels. Exponentiating results makes the values easier to compare.

It is widely used in GAN evaluation to measure the objectiveness and diversity of the generated images. The higher the IS, the better the generated images. The original MirrorGAN paper also takes IS to evaluate the model [2]. It is very suitable for calibrating system performance because it takes both authenticity and diversity into consideration. However, the problem of the metric is that according to our experience, it does not match the visual inspection results. One model may give the higher IS when generating relatively worse images. The score is limited by what the Inception (or other network) classifier can detect, which is directly linked to the training data (commonly ILSVRC 2014).

2) Fréchet Inception Distance: Fréchet Inception Distance (FID) was first introduced in *GANs Trained by a Two Time-Scale Update Rule Converge to Local Nash Equilibrium* in 2017 [5]. Fréchet Inception Distance takes both real and generated images to measure the distance between them. The lower the FID is, the closer the generated images is to real images.

Let $p_w(.)$ be the probability of observed real world data, and $p(.)$ be the probability of generated model data. The equality $p(.) = p_w(.)$ holds except for a non-measurable set if and only if $\int p(.)f(x)dx = \int p_w(.)f(x)dx$ for a basis $f(.)$ spanning the function space in which $p(.)$ and $p_w(.)$ live. These equalities of expectations are used to describe distributions by moments or cumulants, where $f(x)$ are polynomials of the data x. We generalize these polynomials by replacing x by the coding layer of an inception model in order to obtain vision-relevant features. We only consider the first two moments: mean and covariance. The Gaussian is the maximum entropy distribution for given mean and covariance, therefore we assume the coding units to follow a multidimensional Gaussian. The difference of two Gaussians (synthetic and real-world images) is measured by the Fréchet distance also known as Wasserstein-2 distance. We call the Fréchet distance $d(.,.)$ between the Gaussian with mean $(\boldsymbol{m}, \boldsymbol{C})$ obtained from $p(.)$ and the Gaussian with mean $(\boldsymbol{m}_w, \boldsymbol{C}_w)$ obtained from $p_w(.)$ the "Fréchet Inception Distance" (FID), which is given by [5]

$$d^2\left((\boldsymbol{m}, \boldsymbol{C}), (\boldsymbol{m}_w, \boldsymbol{C}_w)\right) = \|\boldsymbol{m} - \boldsymbol{m}_w\|_2^2 + \mathrm{Tr}\left(\boldsymbol{C} + \boldsymbol{C}_w - 2\left(\boldsymbol{C}\boldsymbol{C}_w\right)^{1/2}\right)$$

In that paper, the author also shows that FID is more consistent with the noise level than the Inception Score.

FID is developed based on inception score, which removes noise to a large extent. Also, it becomes popular since 2020. Therefore, FID is also suitable for calibrating our system performance. However, the disadvantage is it only measures the authenticity, namely how close are generated images to real images, but fails to measure the diversity of generated images. Therefore, in our project, we combine both IS and FID to evaluate our results.

E. Results

1) Models for Evaluation: In the result part, we would discuss three different models: AttnGAN, MirrorGAN and MirrorGAN with BERT. MirrorGAN is the model we refer to, which is served as the baseline model. MirrorGAN with BERT is our model. AttnGAN, as the basis of MirrorGAN, is also utilized as comparison.

2) Results Comparison: Qualitatively, the sample outputs of three differents models on five different generated captions is shown as below.

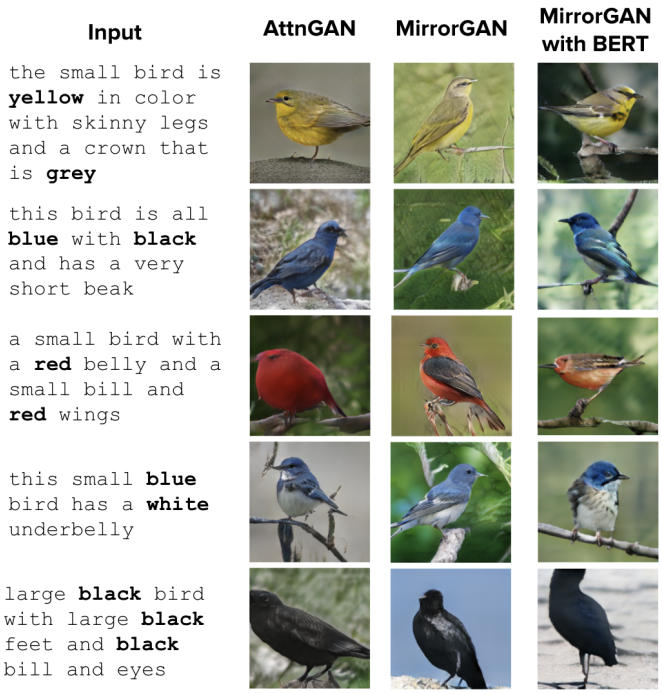

Fig. 8. Sample Outputs for 3 Different Models

The generated captions are listed on the left, while the generated images out of three models are shown on the right. As we can see from the samples, all of three models perform well on semantic consistency (e.g. all satisfying yellow bird, skinny legs and grey crown for the first example). However, our model, mirrorGAN with bert seems lose some authenticity on background (e.g. there are some twist on the first example).

As for the quantitative result, the IS and FID of AttnGAN, MirrorGAN and MirrorGAN with BERT are shown in the table below.

TABLE II
IS AND FID FOR 3 DIFFERENT MODELS

Model	IS	FID
AttnGAN	3.12 ± 0.14	**15.479**
MirrorGAN Reproduction	3.44 ± 0.33	20.013
MirrorGAN with BERT	**3.54 ± 0.18**	30.338

3) Discussion: Our model mirrorGAN with BERT has the highest inception score of 3.54. However, the models performs the worst in FID (higher FID means more far away from real images). Since IS measures both the authenticity and variety of the generated images, while FID only measures the authenticity. We can know that our model may generate diverse images with relatively lower authenticity. This is also align with our qualitative observation that some of our images are twisted at the background. We haven't tried other variants, but we have found the potential reason. Originally, the text encoder is trained based on the specific dataset. Now we use a more general pre-trained model BERT instead. This adds generality, which adds to the variety of the images. However, the increase on generality means less focus on the current dataset, which leads to the drop in authenticity.

V. Conclusion and Insights

In conclusion, we innovatively replace the text encoder of MirrorGAN with BERT in the project. Our model generalizes better using BERT. Originally, mirrorGAN has to train a text encoder on the corresponding dataset, but now we use pretrained bert instead. Therefore, it also reduces the training time. What's more, we improve the diversity of generated images proven by better IS. However, due to limitation on computational resources, our modules are not jointly optimized. And our worst FID indicates the loss of authenticity. Also, as all GAN models, we inspect a very unstable performance. Some potential future improvement may be fine-tuning bert on the dataset to optimize the model and exploring new ways of generating authentic background details.

VI. Appendix

The code is available at https://github.com/item084/eecs592-proj .

References

[1] S. Reed, Z. Akata, X. Yan, L. Logeswaran, B. Schiele, and H. Lee., "Generative Adversarial Text to Image Synthesis," *Proc. Int. Conf. Mach. Learn. Appl.*, vol. 2016, pp. 1060-1069, 2016.

[2] T. Qiao, J. Zhang, D. Xu, and D. Tao, "MirrorGAN: Learning Text-To-Image Generation by Redescription," *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2019.

[3] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova,"BERT: Pre-training of deep bidirectional Transformers for language understanding," *arXiv [cs.CL]*, 2018.

[4] S. Bankar, and S. Ket., "An Analysis of Text-to-Image Synthesis," *Proceedings of the International Conference on Smart Data Intelligence (ICSMDI 2021)*, 2021.

[5] M. Heusel, H. Ramsauer, T. Unterthiner, B. Nessler, and S. Hochreiter. GANs Trained by a Two Time-Scale Update Rule Converge to Local Nash Equilibrium. In Advances in Neural Information Processing Systems (NIPS), 2017.

[6] T. Xu et al., "AttnGAN: Fine-grained text to image generation with attentional generative adversarial networks," *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2018.

[7] S. Naveen, M. S. S. Ram Kiran, M. Indupriya, T. V. Manikanta, and P. V. Sudeep, "Transformer models for enhancing AttnGAN based text to image generation," *Image Vis. Comput.*, vol. 115, no. 104284, p. 104284, 2021.

[8] C. Wah, S. Branson, P. Welinder, P. Perona, and S. Belongie. "The caltech-ucsd birds-200-2011 dataset," *California Institute of Technology*, 2011.

[9] S. Reed, Z. Akata, H. Lee, and B. Schiele., "Learning Deep Representations of Fine-Grained Visual Descriptions," *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

[10] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen. Improved techniques for training gans. In *Advances in Neural Information Processing Systems (NIPS)*, 2016

[11] K. Papineni, S. Roukos, T. Ward, and W. Zhu., "Bleu: a method for automatic evaluation of machine translation," *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, pp. 311-318. 2002.

[12] J. Zhu, T. Park, P. Isola, and A. Efros., "Unpaired image-to-image translation using cycle-consistent adversarial networks." *Proceedings of the IEEE international conference on computer vision*, pp. 2223-2232. 2017.

[13] O. Vinyals, A. Toshev, S. Bengio, and D. Erhan., "Show and tell: A neural image caption generator," *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 3156-3164. 2015.
[14] T. Salimans, I. Goodfellow, W. Zaremba, V. Cheung, A. Radford, and X. Chen. Improved techniques for training gans. In Advances in Neural Information Processing Systems (NIPS), 2016.
[15] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, and Z. Wojna, "Rethinking the inception architecture for computer vision," 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016.
[16] MacKay, David J.C. (2003). Information Theory, Inference, and Learning Algorithms (First ed.). Cambridge University Press. p. 34. ISBN 9780521642989.
[17] https://github.com/qiaott/MirrorGAN
[18] https://github.com/yunjey/pytorch-tutorial/tree/master/tutorials/03-advanced/image_captioning
[19] https://cloud.google.com/translate/automl/docs/evaluate#bleu